CMPT 373
Software Development Methods

Building Software

Nick Sumner
wsumner@sfu.ca
Some materials from Shlomi Fish & Kitware

What does it mean to build software?

- How many of you know how to ***build*** software?

What does it mean to build software?

- How many of you know how to ***build*** software?
- What does the process include?

What does it mean to build software?

- How many of you know how to ***build*** software?
- What does the process include?
- How many of you have heard terms like
  - Build Engineering?
  - Build Configuration?
  - Build Automation?
  - Dependency Management?
  - Continuous Integration?

What does it mean to build software?

- How many of you know how to ***build*** software?
- What does the process include?
- How many of you have heard terms like
  - Build Engineering?
  - Build Configuration?
  - Build Automation?
  - Dependency Management?
  - Continuous Integration?

Just getting something to compile in a repeatable way can be nontrivial

What does it mean to build software?

- Building software includes (at least):
  - version control integration

What does it mean to build software?

- Building software includes (at least):
  - version control integration
  - identifying dependencies & their versions

What does it mean to build software?

- Building software includes (at least):
  - version control integration
  - identifying dependencies & their versions
  - configuring build commands for different build modes & environments

What does it mean to build software?

- Building software includes (at least):
  - version control integration
  - identifying dependencies & their versions
  - configuring build commands for different build modes & environments
  - writing instructions for how to configure & build

What does it mean to build software?

- Building software includes (at least):
  - version control integration
  - identifying dependencies & their versions
  - configuring build commands for different build modes & environments
  - writing instructions for how to configure & build
  - test configuration & execution (performance & correctness)

What does it mean to build software?

- Building software includes (at least):
  - version control integration
  - identifying dependencies & their versions
  - configuring build commands for different build modes & environments
  - writing instructions for how to configure & build
  - test configuration & execution (performance & correctness)
  - automated code quality checking

What does it mean to build software?

- Building software includes (at least):
  - version control integration
  - identifying dependencies & their versions
  - configuring build commands for different build modes & environments
  - writing instructions for how to configure & build
  - test configuration & execution (performance & correctness)
  - automated code quality checking
  - scalable the compilation & linking

What does it mean to build software?

- Building software includes (at least):
  - version control integration
  - identifying dependencies & their versions
  - configuring build commands for different build modes & environments
  - writing instructions for how to configure & build
  - test configuration & execution (performance & correctness)
  - automated code quality checking
  - scalable the compilation & linking
  - possibly even deployment

What does it mean to build software?

- Building software includes (at least):
  - version control integration
  - identifying dependencies & their versions
  - configuring build commands for different build modes & environments
  - writing instructions for how to configure & build
  - test configuration & execution (performance & correctness)
  - automated code quality checking
  - scalable the compilation & linking
  - possibly even deployment
- It is the foundation of getting anything done.

What does it mean to build software?

- How many of you know how to ***build*** software?

What does it mean to build software?

- How many of you know how to ***build*** software?
- You should at least ask yourself:
  - What tools do you use?
  - What workflow?
  - What benefits do you get?
  - What are the painful points?
  - Why haven't you made them less painful?

Build Systems

- Real projects involve many dependent components

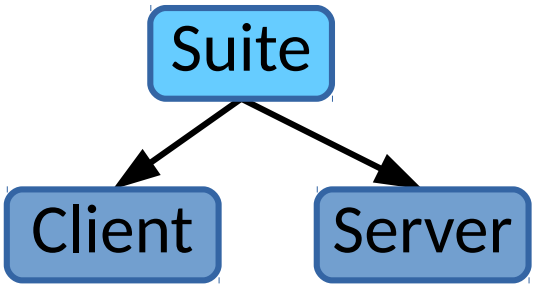
Build Systems

- Real projects involve many dependent components

Build Systems

- Real projects involve many dependent components

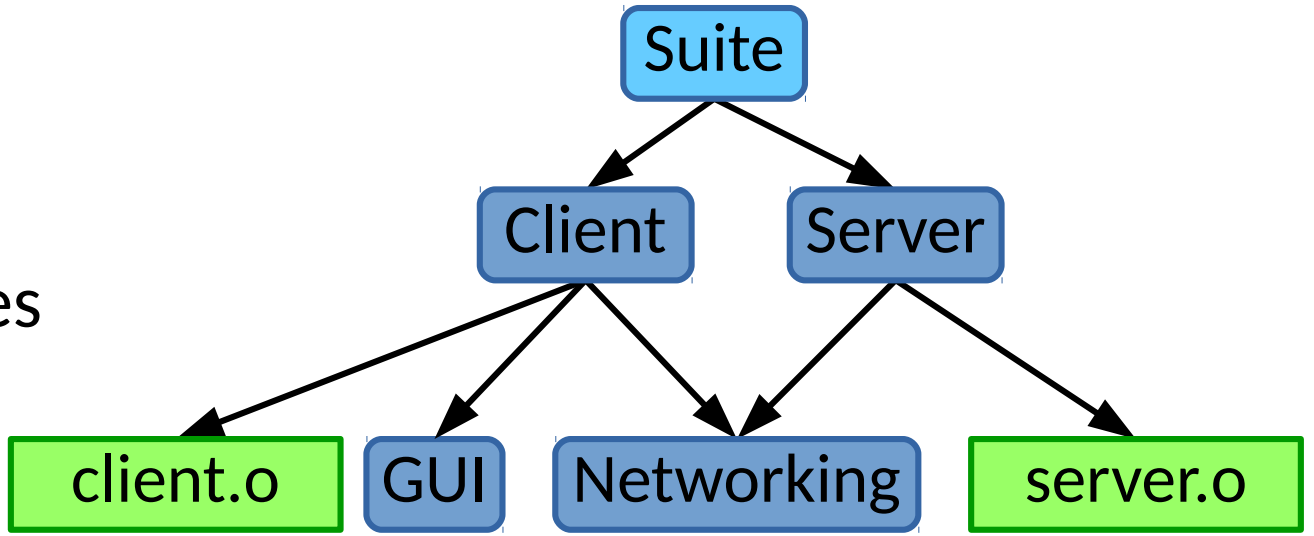
different programs



Build Systems

- Real projects involve many dependent components

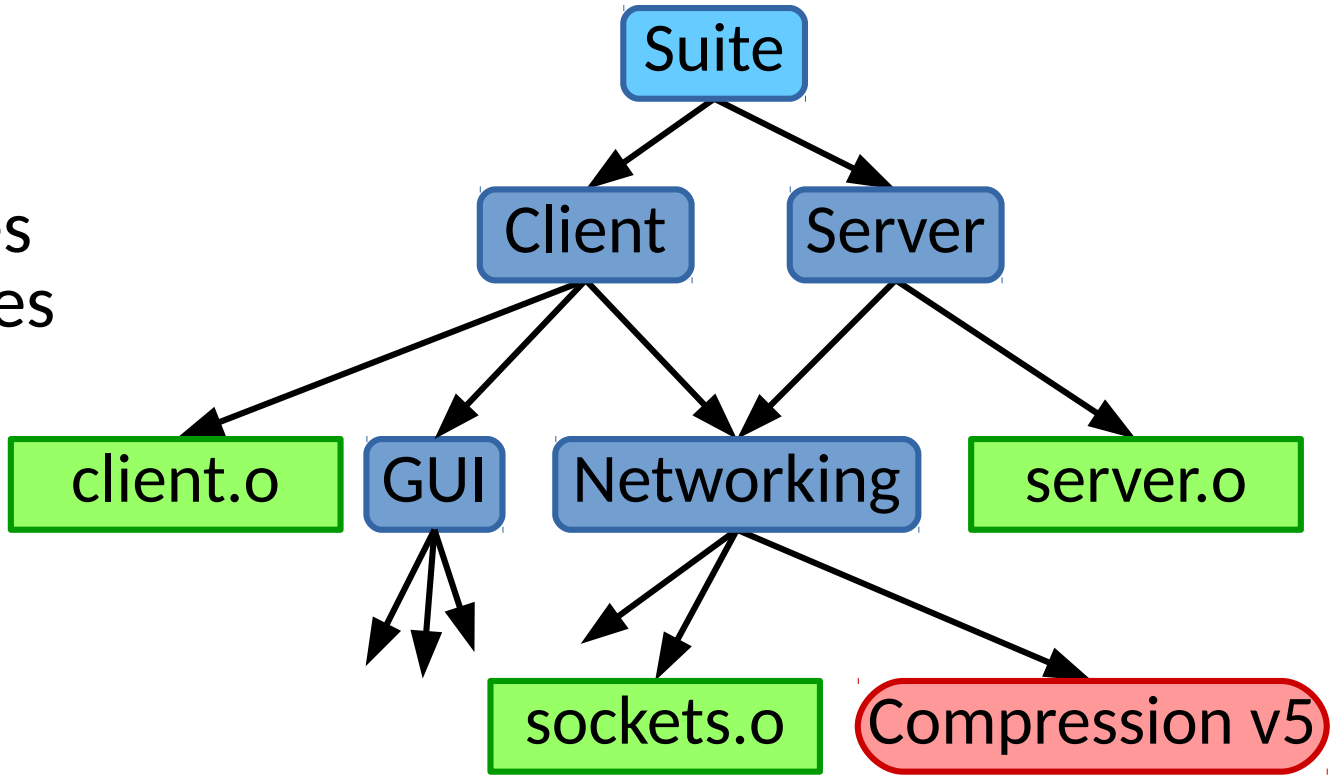
object files
& internal libraries



Build Systems

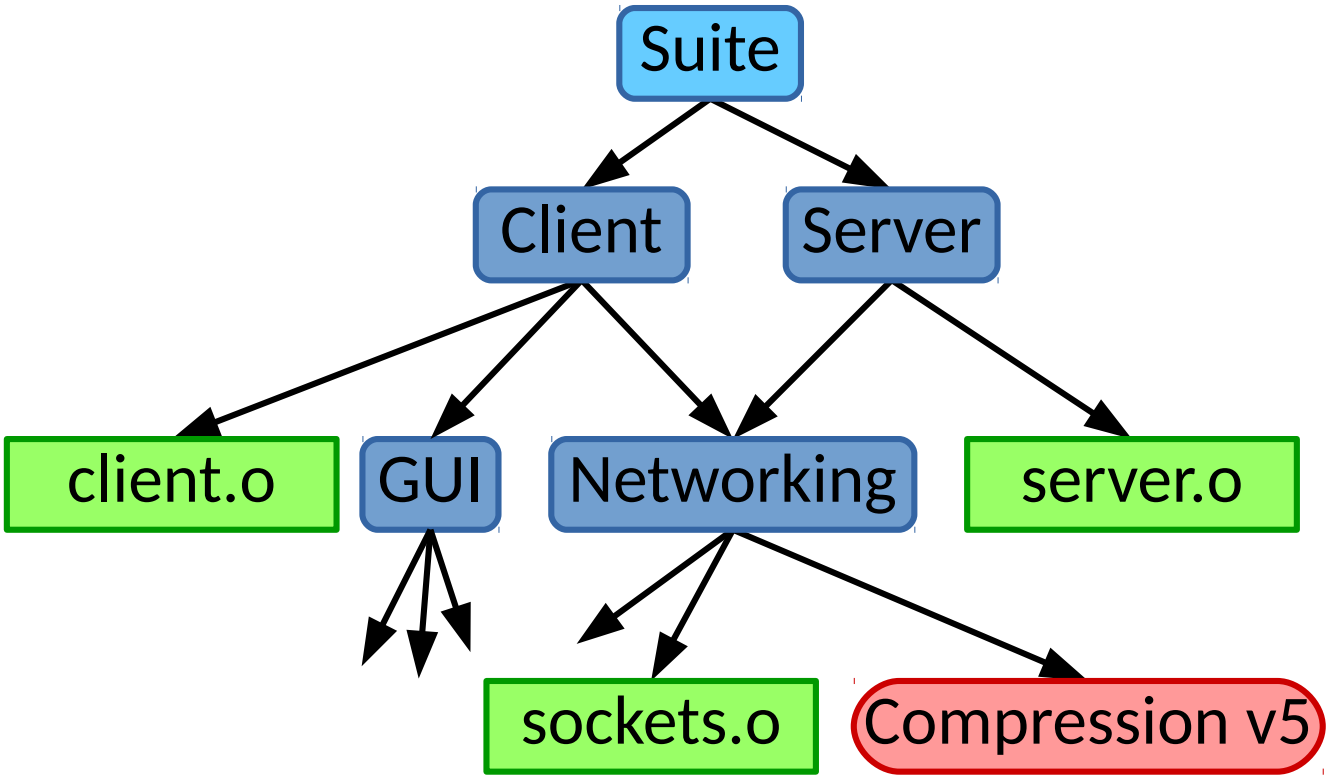
- Real projects involve many dependent components

nested object files
& external libraries



Build Systems

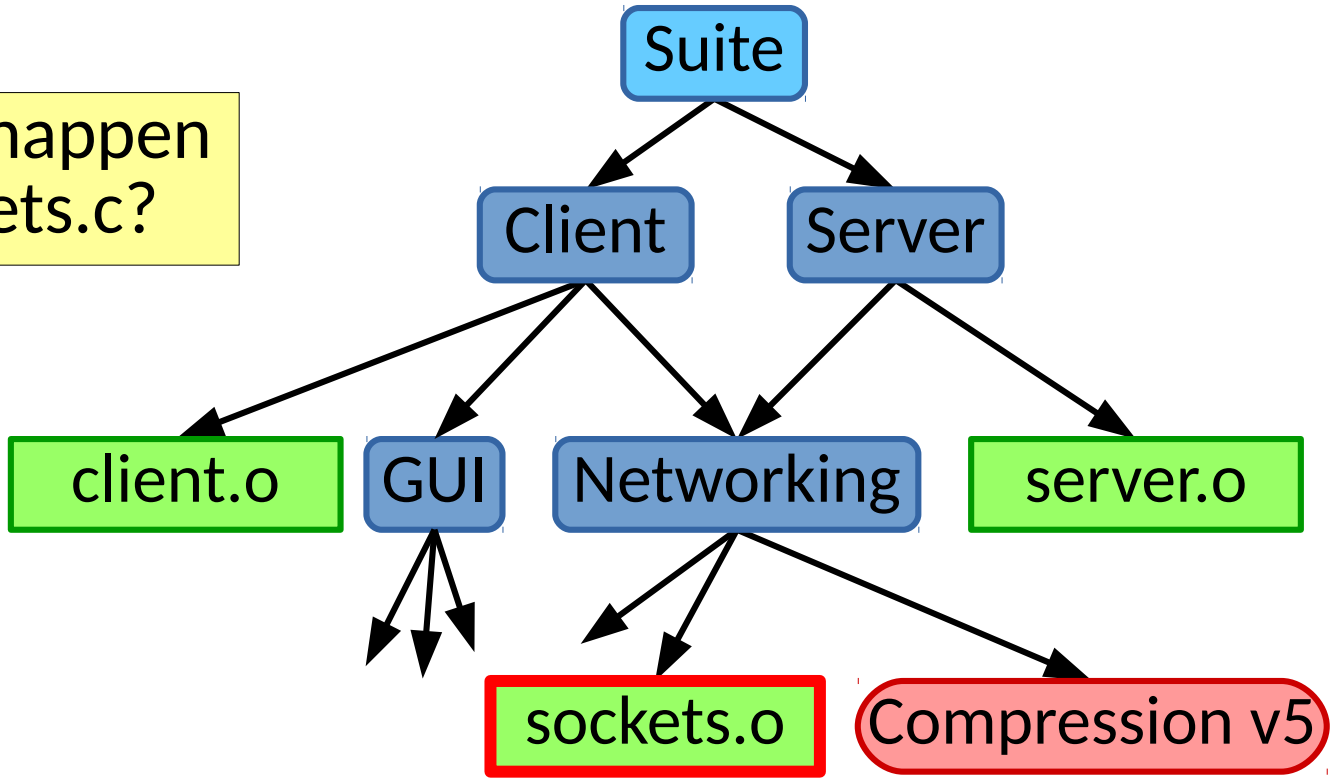
- Real projects involve many dependent components



Build Systems

- Real projects involve many dependent components

What do we want to happen after changing sockets.c?

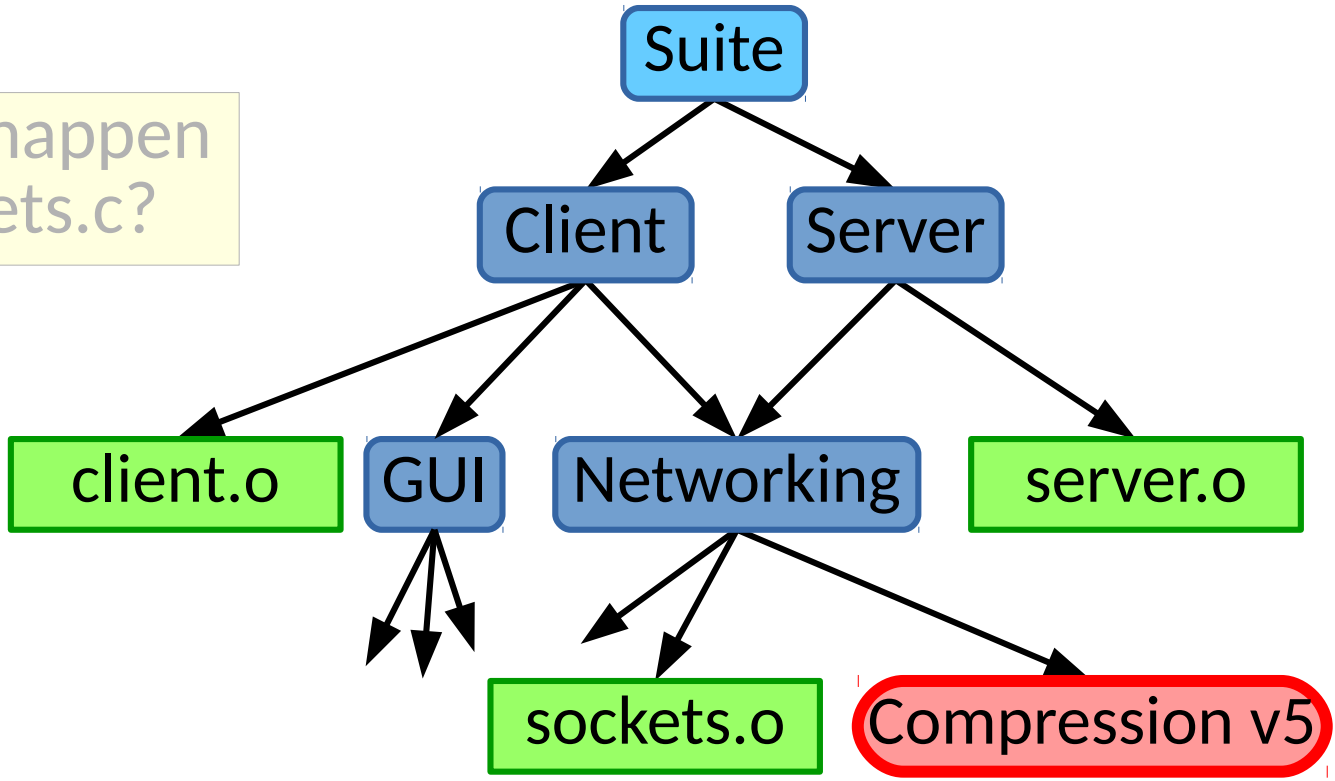


Build Systems

- Real projects involve many dependent components

What do we want to happen after changing sockets.c?

What if we only have version 4 of the compression library?

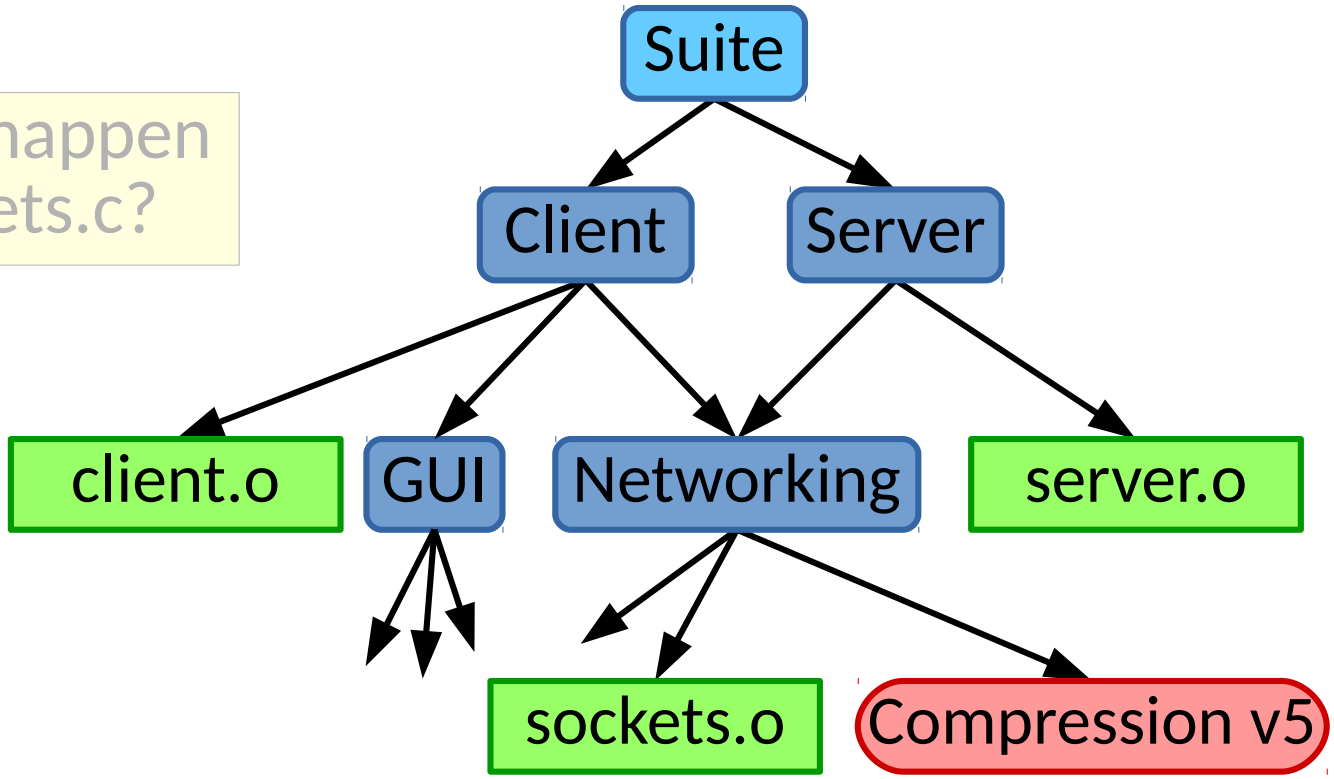


Build Systems

- Real projects involve many dependent components

What do we want to happen after changing sockets.c?

What if we only have version 4 of the compression library?



What if our project requires support for C11?

Build Systems

- Real projects involve many dependent components
- Builds should be
  - Repeatable (Deterministic)

Build Systems

- Real projects involve many dependent components
- Builds should be
  - Repeatable (Deterministic)
  - Incremental

Build Systems

- Real projects involve many dependent components
- Builds should be
  - Repeatable (Deterministic)
  - Incremental
  - Fail fast

Build Systems

- Real projects involve many dependent components
- Builds should be
  - Repeatable (Deterministic)
  - Incremental
  - Fail fast
  - Adapted to the platform/environment

Build Systems

- Real projects involve many dependent components
- Builds should be
  - Repeatable (Deterministic)
  - Incremental
  - Fail fast
  - Adapted to the platform/environment
- Build Systems – build automation tools
  - Identify dependencies & requirements

Build Systems

- Real projects involve many dependent components
- Builds should be
  - Repeatable (Deterministic)
  - Incremental
  - Fail fast
  - Adapted to the platform/environment
- Build Systems – build automation tools
  - Identify dependencies & requirements
  - Automate the building of components

What will be be using?

- CMake
  - Cross-platform build management tool
  - Used by large projects like KDE, Wireshark, LLVM, ...

What will be be using?

- CMake
  - Cross-platform build management tool
  - Used by large projects like KDE, Wireshark, LLVM, …
- What does it do?
  - Given a specification & configuration of your project, CMake creates the build commands for you
  - Analogous to autoconf (but easier to use)

What will be be using?

- CMake
  - Cross-platform build management tool
  - Used by large projects like KDE, Wireshark, LLVM, ...
- What does it do?
  - Given a specification & configuration of your project, CMake creates the build commands for you
  - Analogous to autoconf (but easier to use)

[DEMO]

What does this add?

- Why not just write makefiles manually?

What does this add?

- Why not just write makefiles manually?
  - May need different makefiles for different…

What does this add?

- Why not just write makefiles manually?
  - May need different makefiles for different
    - Operating Systems
    - Compilers
    - Libraries
    - Build Modes
    - …

What does this add?

- Why not just write makefiles manually?
  - May need different makefiles for different
    - Operating Systems
    - Compilers
    - Libraries
    - Build Modes
    - …
  - May need different source files for different “”

What does this add?

- Why not just write makefiles manually?
  - May need different makefiles for different
    - Operating Systems
    - Compilers
    - Libraries
    - Build Modes
    - …
  - May need different source files for different “”
  - Specification can clearly capture
    - Libraries, versions, & even how to download them automatically
    - Semantics of compilation & how to use in analysis tools

What does this add?

- Why not just write makefiles manually?
  - May need different makefiles for different
    - Operating Systems
    - Compilers
    - Libraries
    - Build Modes
    - …
  - May need different source files for different “”
  - Specification can clearly capture
    - Libraries, versions, & even how to download them automatically
    - S… …tools

Build configuration helps manage complex build processes deterministically

What does this add?

- Scalability
  - Replace “make” with analogous scalable tools (“ninja”)

What does this add?

- Scalability
  - Replace “make” with analogous scalable tools (“ninja”)
- Easier tool integration
  - CMake can export compilation rules for other tools

What does this add?

- Scalability
  - Replace “make” with analogous scalable tools (“ninja”)
- Easier tool integration
  - CMake can export compilation rules for other tools

[DEMO]

Preliminary: Out of source builds

- A common bad habit is “in source” building

Preliminary: Out of source builds

- A common bad habit is “in source” building
  - Why is this bad?

Preliminary: Out of source builds

- A common bad habit is “in source” building
  - Why is this bad?
  - May need multiple builds at once: debug, release, …
  - Pollutes version control
  - Makes clean builds complicated

Preliminary: Out of source builds

- A common bad habit is “in source” building
  - Why is this bad?
  - May need multiple builds at once: debug, release, …
  - Pollutes version control
  - Makes clean builds complicated
- Use “out of source” builds instead

Using CMake

- CMakeLists.txt
  - A script in every directory of your project that controls how to build “things” in that directory

Using CMake

- CMakeLists.txt
  - A script in every directory of your project that controls how to build “things” in that directory
- Simple syntax
  - Case insensitive commands

```
command( argument1 argument2 argument3 ...)
```

  - Let's revisit demo 1!

Targets & Commands

- CMake allows you to specify *targets*
  - Executables, libraries, “objects”

```
add_executable(helloworld hellowworld.cpp)
add_library(hellohelper hellohelper.cpp)
```

Targets & Commands

- CMake allows you to specify *targets*
  - Executables, libraries, “objects”

```
add_executable(helloworld hellowworld.cpp)
add_library(hellohelper hellohelper.cpp)
```

- And *commands* that can describe how to build those targets
  - Automatic for executable & library

Targets & Commands

- CMake allows you to specify *targets*
  - Executables, libraries, “objects”

```
add_executable(helloworld hellowworld.cpp)
add_library(hellohelper hellohelper.cpp)
```

- And *commands* that can describe how to build those targets
  - Automatic for executable & library
  - `add_custom_command()` can build others
    - Documentation
    - Media

Directories

- Specify to look for build scripts in subdirectories

```
add_subdirectory(tools)
```

Directories

- Specify to look for build scripts in subdirectories

```
add_subdirectory(tools)
```

- Specify search paths for header files or libraries

```
include_directories(include)
link_directories(lib)
```

Using libraries

- Specify that a specific target needs a library

```
target_link_libraries(target
  lib1 lib2 lib3 ...
)
```

Using libraries

- Specify that a specific target needs a library

```
target_link_libraries(target
  lib1 lib2 lib3 ...
)
```

- How might convenient library use affect program structure and design?
  - How might it help us begin to handle complexity?

General project management

- Specifying project properties
  - Define a project to access variables that control that project

    ```
    project(projectname)
    ```

General project management

- Specifying project properties
  - Define a project to access variables that control that project

    ```
    project(projectname)
    ```
- Print information out during the build process

  ```
  message("Built with flags: ${CMAKE_CXX_FLAGS}")
  ```

General project management

- Specifying project properties
  - Define a project to access variables that control that project

    ```
    project(projectname)
    ```
- Print information out during the build process

  ```
  message("Built with flags: ${CMAKE_CXX_FLAGS}")
  ```
- Controlling where things are built

  ```
  set(CMAKE_RUNTIME_OUTPUT_DIRECTORY
      "${PROJECT_BINARY_DIR}/bin")
  set(CMAKE_LIBRARY_OUTPUT_DIRECTORY
      "${PROJECT_BINARY_DIR}/lib")
  ```

General project management

- Finding a resource that you need to use

  `find_package(externalproject)`

  `find_library(library)`

General project management

- Finding a resource that you need to use

```
find_package(externalproject)
find_library(library)
```

- Installation

```
install(TARGETS target1 target2 ...
  DESTINATION /tmp/
)
```

Control structures

- IF

```
if(condition)
elsif(condition2)
else()
endif()
```

Control structures

- IF

```
if(condition)
elsif(condition2)
else()
endif()
```

- Looping

```
foreach(loop_var arg1 arg2 ...)
  command(${loop_var})
endforeach(loop_var)
while(condition)...
```

Control structures

- IF

```
if(condition)
elsif(condition2)
else()
endif()
```

- Looping

```
foreach(loop_var arg1 arg2 ...)
  command(${loop_var})
endforeach(loop_var)
while(condition)...
```

- Functions

```
function(function_name arg1 arg2 ...)
  command(${arg1})
endFunction(function_name)
```

Examples

- Let's take a look at some examples....